Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Variable Naming Impact on AI Code Completion: An Empirical Study

Alexey Yakubov

alexey.yakubov@gmail.com

Research Article

Keywords: Code completion, Large language models, AI code generation, Variable naming, Programming productivity, LLM evaluation

Posted Date: July 23rd, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7180885/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Variable Naming Impact on AI Code Completion: An Empirical Study

Alexey Yakubov
alexey.yakubov@gmail.com

July 21, 2025

1 Abstract

As AI code completion tools become central to software development, a fundamental question emerges: do variable naming conventions that aid human comprehension also improve AI model performance? We investigate this question using a controlled experimental design with 500 Python code examples generated by mistralai/Magistral-Small-2506 (24B parameters, quantized to 8bit), transformed into 7 naming schemes (descriptive, minimal, obfuscated, original, PascalCase, SCREAM_SNAKE_CASE, snake_case), and tested across 8 models (0.5B-8B parameters) spanning two architectures (Llama and Qwen). The same model performs renaming transformations and serves as a semantic judge for evaluating completion outputs. Despite requiring more tokens, descriptive variable names consistently achieved the best semantic similarity (0.874), while obfuscated names performed worst (0.802) — consistent with human cognition research. Our evaluation combines exact token matching, Levenshtein similarity, and semantic similarity. Strong correlations between syntactic and semantic metrics (r=0.945) validate our evaluation approach. Model scaling effects demonstrate clear performance improvements with size, with semantic similarity ranging from 0.743 (1B) to 0.898 (7B) for Llama models. These findings provide initial quantitative evidence for code style considerations in AI code completion, with implications for developers using AI coding assistants.

2 Introduction

While extensive research has established that variable naming conventions significantly impact human code comprehension [1, 2, 3], their impact on AI code completion remains understudied. This gap becomes increasingly important as AI-powered tools reshape software development practices.

This work quantifies how different naming conventions affect completion accuracy across multiple model sizes and architectures, using LLM-as-judge evaluation to assess semantic similarity alongside traditional metrics. We use Magistral-24B as both generator and evaluator to ensure consistent assessment criteria. Since our focus is on how different models complete code under varying naming schemes, using a consistent judge eliminates evaluation variance and provides stable ground truth for semantic correctness. The strong correlation between semantic judgments and syntactic metrics (r=0.945) suggests our findings reflect genuine code completion patterns rather than model-specific artifacts.

3 Related Work

Recent advances in tokenization research show that semantic clarity, not compression efficiency, drives AI performance [8, 9]. Code completion evaluation frameworks like L2CEval [7] demonstrate clear scaling laws across programming tasks, establishing the foundation for systematic evaluation of naming convention effects. Building on human factors research showing that descriptive names improve accuracy and reduce completion time [1, 2, 3], we investigate whether similar patterns emerge in AI systems.

4 Methodology

- **Dataset:** 500 Python code samples generated by Magistral-24B, transformed into 7 naming conventions (descriptive, minimal, obfuscated, original, PascalCase, SCREAM_SNAKE_CASE, snake_case) using the same model for consistency.

- **Models:** 8 models spanning 0.5B-8B parameters across Llama and Qwen architectures, enabling scaling analysis and cross-architectural comparisons. Our evaluation focuses on models up to 8B parameters due to computational constraints. Validation with larger models (70B+) represents important future work, though our cross-architecture consistency suggests the core findings may generalize.

- **Task:** Complete exactly 25 tokens after truncating the last 25 tokens from each code sample. Models use their native tokenizers for consistent evaluation.

- **Metrics:** Exact match rate, Levenshtein similarity, and semantic similarity assessed by Magistral-24B judge focusing on algorithmic logic and functional equivalence. Semantic similarity is scored on a 0-1 scale similar to Levenshtein similarity, with detailed reasoning for each assessment.

- **Analysis:** One-way ANOVA, Cohen's d effect sizes, and correlations between evaluation metrics across 500 base samples with 7 naming schemes and 8 models.

- **Generator/Judge**: Magistral-24B was selected based on several methodological constraints and requirements. First, we evaluated models from two distinct architectures (LLaMA and Qwen), precluding the use of generator models from these same families to reduce architectural bias. Second, all experiments were conducted locally to ensure reproducibility and control computational costs, eliminating proprietary models from OpenAI and Anthropic. Given these constraints, Magistral-24B offered the optimal balance between code generation capability and assessment quality while maximizing utilization of available hardware (32GB VRAM).

5 Results

The bar chart below reveals our core finding: descriptive variable names consistently outperform all other naming schemes across models, while obfuscated names perform worst. The side-by-side comparison also demonstrates the strong correlation between syntactic and semantic measures ($r = 0.945$), validating our LLM-as-judge evaluation approach.

Key Findings:

- **Strong metric correlations** validate our approach: Levenshtein vs Semantic ($r = 0.945$), Exact match vs Semantic ($r = 0.884$)

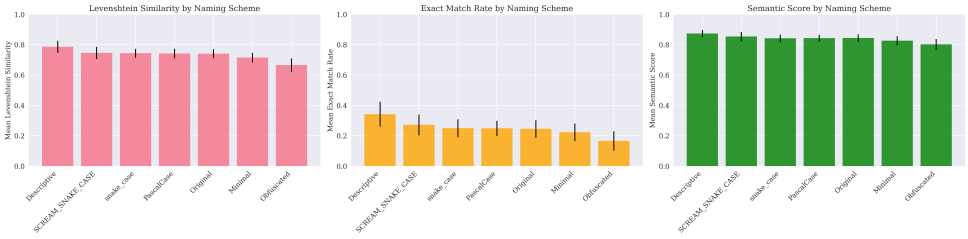


Figure 1: Naming Convention Performance Comparison

- **Clear scaling benefits**: CodeLlama-7B achieves best performance (0.898 semantic score) vs Llama-1B (0.836) for descriptive naming
- **Cross-architectural consistency**: Minimal difference between Llama (73.3%) and Qwen (73.6%) families in Levenshtein similarity
- **Ranking consistency**: descriptive > SCREAM_SNAKE_CASE > snake_case > pascal_case > original > minimal > obfuscated

5.1 Semantic Evaluation Examples

Our semantic similarity assessment focuses on algorithmic logic and functional equivalence rather than syntactic matching. Below are representative examples of our LLM judge reasoning:

High Semantic Similarity (Score: 1.0): For an ISBN validator with descriptive naming completed by Llama-7B: "The AI-completed portion is semantically identical to the original. The algorithmic approach, control flow, and data flow are preserved exactly. The only difference is a minor formatting issue (missing space after 'else'), which doesn't affect functionality."

Medium Semantic Similarity (Score: 0.5): For an image sharpening filter with descriptive naming completed by Llama-7B: "The completion attempts to continue the sharpening operation but introduces a new `cv2.addWeighted` call without completing the previous operation. The original used `cv2.filter2D` with the kernel and factor, while the completion adds another weighted operation which changes the algorithmic approach."

These examples demonstrate our judge's focus on functional correctness over syntactic precision.

5.2 Token Efficiency vs Performance

Despite consuming more tokens, descriptive naming schemes consistently achieve superior completion accuracy, demonstrating that semantic clarity drives AI performance over compression efficiency. The line chart below shows how each model's performance scales with token consumption across naming schemes.

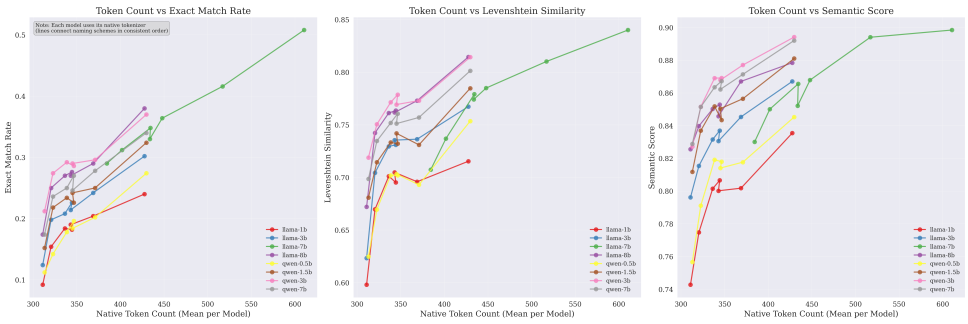


Figure 2: Token vs Performance Line Chart

5.3 Cross-Model Consistency

The performance patterns hold consistently across model architectures and sizes, as shown in the heatmaps below. This cross-architectural consistency suggests these are fundamental properties of code completion rather than model-specific artifacts.

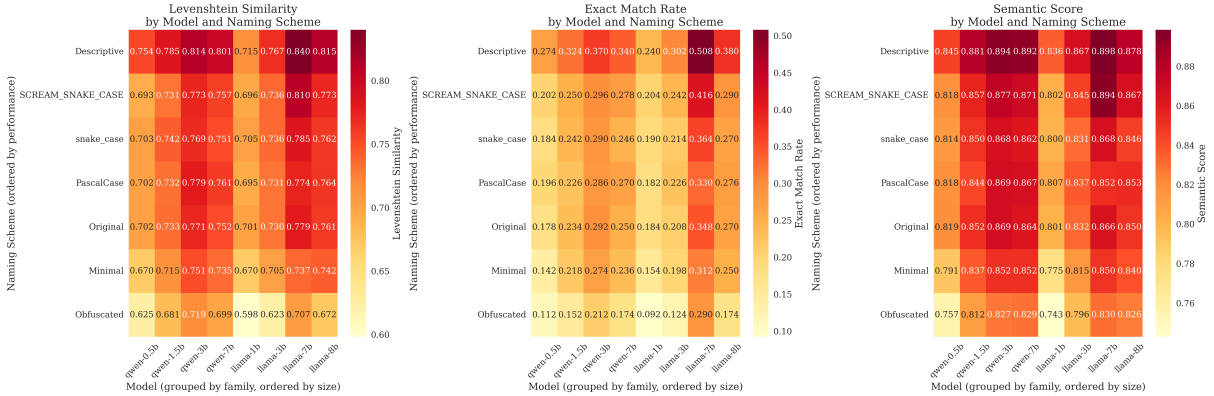


Figure 3: Performance Heatmaps

5.4 Model Performance

CodeLlama-7B achieves the best overall performance (36.7% exact match, 0.866 semantic score), demonstrating the benefit of code specialization. Both model families show consistent scaling effects, with minimal architectural differences (Llama: 73.3% vs Qwen: 73.6% Levenshtein similarity overall). Notably, Qwen-3B slightly outperforms Qwen-7B across all metrics, suggesting that scaling effects may vary by architecture and training approach.

5.5 Naming Convention Impact

Naming Scheme	Levenshtein Similarity	Exact Match	Semantic Score
Descriptive	0.786	34.2%	0.874
SCREAM_SNAKE_CASE	0.746	27.2%	0.854
snake_case	0.744	25.0%	0.842
PascalCase	0.742	24.9%	0.843
minimal	0.715	22.3%	0.826
obfuscated	0.666	16.6%	0.802

These results extend human comprehension findings [2] to AI systems, with semantic evaluation confirming that readable code benefits artificial intelligence across multiple dimensions.

5.6 Statistical Validation

ANOVA results show significant effects across all metrics ($F > 60$, $p < 0.000001$) with meaningful effect sizes. Key comparisons show Cohen's $d = 0.41$-0.48 for descriptive vs obfuscated naming across metrics. Strong correlation between Levenshtein and semantic similarity ($r = 0.945$) validates our LLM-as-judge approach, while consistent effects across architectures suggest fundamental code completion properties rather than model-specific artifacts.

6 Discussion

Our findings extend human factors research [13, 14] to AI systems, showing that AI models exhibit similar preferences for descriptive naming as humans. Performance improvement from descriptive vs obfuscated naming suggests AI models leverage semantic information embedded in variable names, aligning with Schmidt et al.'s [9] finding that semantic clarity drives performance over compression efficiency.

The LLM-as-judge approach provides interpretable semantic evaluation with strong correlation to syntactic measures ($r = 0.945$), validating traditional metrics while reducing bias. Consistent effects across architectures using independent generation suggest these reflect fundamental code completion properties rather than model-specific artifacts.

7 Implications

- **For Developers**: Descriptive variable names require 41% more tokens than obfuscated code but achieve 8.9% better semantic performance, demonstrating that token overhead is justified by measurable completion accuracy gains.

- **For Organizations**: Code style guides should prioritize readability for both human maintainers and AI tools. The 17.6 percentage point improvement suggests potential productivity benefits in AI-assisted development.

- **For Research**: LLM-as-judge methodology offers a scalable approach for semantic code evaluation, with strong correlation to syntactic measures validating traditional metrics while enabling semantic interpretation.

8 Limitations

Our evaluation is limited to 500 Python code samples and 25-token completion tasks. The dataset generated by Magistral-24B may introduce systematic patterns, though cross-architecture validation suggests robustness. Model coverage is limited to 0.5B-8B parameters, not covering larger state-of-the-art models. We lack controls for code complexity, domain-specific effects, and baseline human performance comparisons. Our token-level matching evaluation, while augmented by semantic similarity, cannot capture execution-based correctness — future work should explore functional validation. Results may not generalize to domain-specific code or complex software systems.

9 Conclusion

This study suggests that variable naming conventions significantly impact AI code completion performance, with descriptive naming achieving better accuracy than obfuscated alternatives. Using LLM-as-judge methodology, we show strong correlation between syntactic and semantic measures ($r = 0.945$), validating traditional metrics while reducing bias during evaluation.

Key contributions include: (1) providing evidence that semantic clarity drives AI performance over compression efficiency, (2) establishing clear performance hierarchy for preliminary guidelines, and (3) showing consistent effects across model architectures, suggesting fundamental code completion properties.

These findings provide initial quantitative evidence for code style considerations in AI code completion, with implications for Python-based AI-assisted development. Future work should explore additional programming languages, larger models, and repository-level contexts.

References

[1] Lawrie, D., Morrell, C., Feitelson, D., & Binkley, D. (2006). What's in a name? A study of identifiers. In Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC'06).

[2] Feitelson, D. G., Mizrahi, M., Noy, N., Ben Shabat, A., Barad, O., & Mordechai, S. (2017). How developers choose names. In Proceedings of the 39th International Conference on Software Engineering (ICSE'17).

[3] Schankin, A., Berger, A., Holt, D. V., Hofmeister, J. C., Peitek, N., & Brechmann, A. (2018). Descriptive compound identifier names improve source code comprehension. In Proceedings of the 26th Conference on Program Comprehension (ICPC'18).

[4] Butler, S., Wermelinger, M., Yu, Y., & Sharp, H. (2010). Exploring the influence of identifier names on code quality: An empirical study. In Proceedings of the 14th European Conference on Software Maintenance and Reengineering (CSMR'10).

[5] Hofmeister, J., Siegmund, J., & Holt, D. V. (2017). Shorter identifier names take longer to comprehend. In Proceedings of the 24th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER'17).

[6] Ciniselli, M., Cooper, N., Pascarella, L., Mastropaolo, A., Aghajani, E., Poshyvanyk, D., Di Penta, M., & Bavota, G. (2021). An empirical study on the usage of transformer models for code completion. arXiv preprint arXiv:2108.01585.

[7] Ni, A., Yin, P., Zhao, Y., Riddell, M., Feng, T., Shen, R., Yin, S., Liu, Y., Yavuz, S., Xiong, C., Joty, S., Zhou, Y., Radev, D., & Cohan, A. (2023). L2CEval: Evaluating language-to-code generation capabilities of large language models. arXiv preprint arXiv:2309.17446.

[8] Sun, K., Qi, P., Zhang, Y., Liu, L., Wang, W. Y., & Huang, Z. (2023). Tokenization consistency matters for generative models on extractive NLP tasks. arXiv preprint arXiv:2212.09912.

[9] Schmidt, C. W., Reddy, V., Zhang, H., Alameddine, A., Uzan, O., Pinter, Y., & Tanner, C. (2024). Tokenization is more than compression. arXiv preprint arXiv:2402.18376.

[10] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... & Zaremba, W. (2021). Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374.

[11] Tong, W., & Zhang, T. (2024). CodeJudge: Evaluating code generation with large language models. arXiv preprint arXiv:2410.02184.

[12] Avidan, E., & Feitelson, D. G. (2017). Effects of Variable Names on Comprehension: An Empirical Study. Proceedings of the 25th International Conference on Program Comprehension (ICPC), 55-65.

[13] Arnaoudova, V., Di Penta, M., & Antoniol, G. (2015). Linguistic antipatterns: What they are and how developers perceive them. Empirical Software Engineering, 21(1), 104-158.

[14] Sharif, B., & Maletic, J. I. (2010). An eye tracking study on camelCase and under_score identifier styles. In Proceedings of the 18th IEEE International Conference on Program Comprehension (ICPC'10).